Global Path Planning for Mars Rover Exploration

Paul Tompkins
Anthony Stentz
David Wettergreen

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Phone: 412-268-5188
{pauldt, tony, dsw}@ri.cmu.edu

Abstract - TEMPEST is a planner for long-range planetary navigation that bridges the gap between path planning and classical planning and scheduling. In addition to planning routes, our approach yields the timing and placement of actions to conserve and restore expendable resources and that abide by operational constraints. TEMPEST calls upon the Incremental Search Engine (ISE) to enable heuristic path planning and efficient re-planning under global constraints, over a four dimensional state space. We describe our approach, then demonstrate how the planner operates in a simulated Mars science traverse. Following a brief summary of TEMPEST results from a recent rover field experiment, we evaluate our research progress and describe our current and future work.

Table of Contents

1. Introduction[1]

The NASA Mars Technology Program (MTP) [5] currently funds a wide range of technology projects in support of the Mobile Science Laboratory (MSL) mission concept, under investigation for the 2009 launch opportunity [35]. Under this umbrella, the MTP supports a number competitively-selected robotics research tasks. In contrast with more general research calls, the MTP selects technologies based on specific MSL mission needs, and requires that they meet basic criteria for developmental maturity prior to selection. To ease software evaluation and possible inclusion in future missions, participating teams are expected to integrate their software within a new NASA software architecture, CLARAty (Coupled Layer Architecture for Robotic Autonomy) [34], by the conclusion of the project. Participants in the program include teams at NASA/JPL, NASA Field Centers and universities.

As one of the MTP participants, our team is developing software for autonomous global path planning, combining long-range route planning, resource management and enforcement of constraints. Several MSL design features motivate our work. To enable the investigation of specific surface targets, the MSL mission seeks long-range mobility to overcome worst-case landing errors. Furthermore, MSL may involve travel to several distinct sites, interleaving periods of dedicated, localized science data collection with periods of traverse and opportunistic science. To alleviate the power constraints of previous missions, the MSL rover may be powered by RTG (Radioisotope Thermo-electric Generator), and possibly supplemented by solar arrays. However, the power sources may not be sufficient to supply locomotion for long time periods. Batteries will store energy for use in high-power activities. Finally, current plans dictate a primary mission lasting up to 500 sols, entailing daily and seasonal lighting changes. All these factors motivate global path planning.

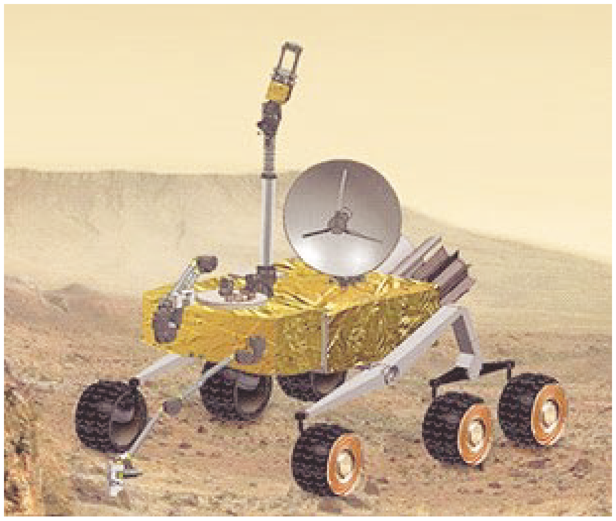


Figure 1: Mobile Science Laboratory Rover (artwork is conceptual and pre-decisional) - courtesy NASA JPL.

Our work in this area began under, and is supplemented by, other NASA-funded projects, namely the Sun-Synchronous Navigation project [36][37][31] and the Life in the Atacama project [38][32]. In this paper, we describe our progress to date, as a result of all these efforts, but with an emphasis on our work directed towards the MTP. We begin by further motivating global path planning for planetary applications, then review the current state of research relevant to the problem. The main body of the paper then details our approach to solving the problem, and demonstrates our software in both

1. 0-7803-8155-6/04/$17.00 © 2004 IEEE
IEEE AC paper #1339

simulation and on rover prototypes in Mars-analog terrain. We conclude by evaluating our current state of research, and by charting the course for continued research and development in the coming years.

2. Motivation

Long-Range Navigation

A critical task in achieving rover autonomy is automatic route planning between a landing site and operations sites. To date, path planning research for planetary rovers has focused on the problem of navigating locally through fields of rock obstacles en route to a global position goal, over tens of meters. In upcoming missions, long-distance and long-term path planning will be necessary for a robot to travel between its landing site and operations sites. The Mobile Science Laboratory mission, slated for 2009, may have a landing error ellipse of 10 kilometers along-track by 3 kilometers cross-track [5]. Accordingly, to target a specific location for scientific study, a robot must be able to traverse 10 kilometers or more of terrain.

Local path planning strategies are tailored specifically for travel amongst rocks at or below the scale of the rover, and only consider terrain within a few meters of the robot to make path decisions. Other sources of data, for example orbital imagery, will enable a rover to anticipate opportunities and hazards and to incorporate these predictions into path selection.

Long distances and durations introduce new complications absent in current local path planning. These include navigation around and through large-scale terrain, the local occlusion of light and communications, and variable lighting induced by planetary rotation.

Resource Management

Resource management is essential to rover self-sufficiency. Resources take many forms, from battery energy and onboard memory, whose state can be defined by a fraction of the total capacity, to shared hardware such as cameras, whose usage state is binary.

The favored approach for autonomous resource management is through AI planning and scheduling, for example the Remote Agent [2] and ASPEN [4] systems. For orbiting spacecraft and short-range rovers, energy expenses are dominated by loads from active electronics components. Solar power follows simple models, and the timing of nightfall and eclipses are easily predicted. In these cases, the traditional planning and scheduling approach is powerful. However, many long-range rover resource expenses and gains are path-dependent and cannot be adequately considered outside a path planner. For example, energy management must consider locomotion energy as a function of terrain, and whether shadows will permit solar charging as a function of position and time. A rover must consider the effects of path choices on energy balance to determine which paths are feasible or optimal.

As mentioned earlier, the MSL rover is likely to incorporate an RTG for continual power, day and night. However, it is probable that the RTG will not provide a sufficient output to power continuous locomotion. The RTG output power and MSL battery capacity will govern the maximum duration for traverses, and will determine how long a battery recharge will take. A planning and scheduling module and a resource-cognizant path planner could share the duty of coordinating the timing of traverses, battery charging and science activities.

3. Related Work

The global path planning problem is multi-faceted - ideally it calls for a solution that solves for optimal trajectories through spatial, temporal and resource dimensions. These must also satisfy local and global operational constraints. Furthermore, an algorithm operating under the uncertainty of planetary exploration must have the ability to efficiently re-plan as a robot gains new information about its environment. As such, we must examine the existing body of work in several areas to adequately assess the current state-of-art.

General Path Planning - At the level of algorithms, a number of approaches address path planning to goals while avoiding obstacles. Most plan in the configuration space, in which each dimension is a degree of freedom for robot motion, and obstacles in the real world map to unreachable regions in the space. Potential field algorithms [12] plan by superposing an attractive field centered at a global goal with repulsive fields surrounding obstacles. A "plan" simply follows the steepest resulting gradient to the goal. However, this strategy is vulnerable to local minima, and hence is incomplete. Many algorithms focus more on region decomposition or "skeletonization", and then derive plans using a simple graph search between regions. These include simple grid approaches, cell decomposition [22], visibility graphs [16], and Voronoi graphs [17]. Randomized algorithms, often used for robots with many degrees of freedom, quickly explore high-dimensional spaces to find feasible paths. These include probabilistic road maps [11] and rapidly-exploring random trees [14]. These algorithms are best suited in applications where finding the optimal path is unnecessary.

Search - Several search algorithms find optimal paths through a graph, according to an objective function, given the cost of actions between nodes in the graph. The A* algorithm uses the concept of "admissible" heuristics to find the optimal path in the minimum number of search steps [19]. However, A* neither enables global constraint satisfaction nor efficient re-planning. The ABC algorithm [15] generalizes A* to handle a variety of global constraint forms. However, since constraints are tracked in each state and non-dominated paths are retained, we expect ABC to exhibit the same complexity as exhaustive search, and hence impractical for online operation. In the direction of re-planning, the D* algorithm offers the first-time planning optimality of A*, but generalizes to enable optimally efficient path repair in response to changing cost information [26][27][28]. Where A* is forced to plan from scratch if any state transition costs change, D* determines which nodes in the graph are affected by the changes, and repairs only those nodes. The effect is a dramatic improvement in speed for localized changes to the costs. However, D* does not enable global constraint satisfaction.

The CD* algorithm is one of the few to address the combined problem of optimal path planning and re-planning under global constraints [29]. It defines a composite function that is a weighted sum of the objective function and a constraint function. To plan initially, CD* uses a binary search to find an optimal weighting, calling D* with different weights until it achieves the optimal constraint-satisfying path. In re-planning, CD* re-uses the graphs created by D* in the initial binary search to find a new weight. Changing the weight in D* is equivalent to re-specifying all the costs in the graph, eliminating the benefits of the D* algorithm. However, in many cases very few of the weights change with updates to the costs, and so CD* is often very efficient. In the worst case, the D* searches perform like A*. As we describe in the Technical Approach section, an algorithm called Incremental Search Engine [30] provides optimal planning and re-planning under global constraints in high-dimensional state spaces, yet with greater predictability in performance than CD*.

Local Terrain Navigation - A large body of work addresses the local terrain navigation problem. The fact that the NASA Mars Exploration Rovers will use stereo-vision based obstacle detection and avoidance [8] in 2004 is testament to its growing maturity. Several related terrestrial systems display similar capabilities for reliable autonomous navigation over tens or hundreds of meters [13][25][27][37], and new techniques incorporating detailed vehicle kinematics and dynamics [33] and estimates of soil properties [10] may achieve vehicle agility over a much wider range of conditions. We view all these developments as supporting the global path planning cause - the better the local problem is solved, the more likely solutions for the global problem become demonstrable.

Global Terrain Navigation - Very little work has addressed natural terrain path planning at the large scale. We attribute this to a sensible prioritization - planetary surface exploration has had no immediate need for global navigation, and realistically, the large-scale problem cannot be solved until reliable solutions exist for the local problem. Richbourg et al. [20] uses optical analogies to solve high-level path planning through polygonal homogeneous cost fields. Rowe [21] extends that work to include linear features like roads and rivers. Pai and Reissell present a multi-resolution method for evaluating terrain elevation models for traversability using wavelets [18], while Howard and Seraji use fuzzy-logic filters [9]. Both techniques could prove valuable for this problem. However, our preference was to coarsely model terrain-vehicle interaction to estimate traversability and locomotion power. These techniques are more heuristic in nature.

Temporal Path Planning - Solutions for temporal path planning often decompose the problem into separate problems of path selection and subsequent path timing. The early manipulation work of Bobrow *et al.* [3] first selects paths that avoid static obstacles, then select acceleration profiles that satisfy dynamics and motor torque constraints. Fraichard adds dynamic obstacles to the problem [6]. Shiller and Gwo apply a related technique for a vehicle rolling on smooth terrain [24]. This general approach is dissatisfying because it avoids the most interesting inter-relationships between path selection and timing. It is also not complete - in certain instances, a selected path may not permit a feasible solution according to dynamics or obstacle constraints.

Resource Management - Research to solve the resource management problem is restricted almost exclusively to classical planning and scheduling. One could argue that, for non-renewable energy resources (e.g. primary batteries, fuel tanks), minimum-distance paths are also optimal energy paths. Unfortunately, this approach won't work for renewable energy resources like rechargeable batteries. Shillcut [23] evaluates several robot coverage patterns in terms of energy cost and solar energy collection, in one of the few examples of path-oriented resource management. However, the work falls short of incorporating these metrics into a planning framework.

In the realm of more classical planning and scheduling approaches, the Remote Agent Experiment [2] and the ASPEN and CASPER systems [4] are perhaps the most clearly space relevant. Each decomposes high-level mission goals into low level sequences of concurrent actions that constitute plans. The selection and ordering of actions achieves the mission goals and satisfies all global and interaction constraints, including those imposed by resources. Planner-scheduler systems perform well under highly-predictable spaceflight conditions, or in situations of limited surface mobility. However, just as path planners tend to avoid resource considerations, planner-scheduler systems largely skirt path issues. None satisfactorily analyze the inter-dependence of activities and paths, for example, how route selection could affect resource availability and consequently enable or prevent certain rover activities. That said, with such highly-developed systems for activity planning, it seems the wisest approach would integrate a path-knowledgeable planner-scheduler with a separate, resource-knowledgeable path planner.

4. Technical Approach

TEMPEST combines a novel search algorithm called Incremental Search Engine with a high-level search strategy to find optimal plans. It defines the global path planning domain with coarse models of the world, the rover, available actions and operational constraints. The following sections describe this technical approach.

Incremental Search Engine (ISE)

ISE is the search algorithm that allows TEMPEST to reason about rover actions, achieving goals efficiently, and satisfying constraints. ISE is a graph-theory based, heuristic search algorithm optimized for planning and re-planning in high-dimensional spaces [30]. The algorithm is complete, and optimal to the resolution of the discrete state space and actions.

An ISE state space comprises two types of variables. IPARMS are independent variables, whose values are discrete and generally coarse. DPARMS are arbitrarily fine dependent variables, whose values are grouped according to equivalence classes. Actions define transitions between IPARMS values. For each action, a user-defined state transition function defines how DPARMS change in response to changes in IPARMS. For example, an application might define two spatial IPARMS variables (X, Y), and a DPARMS variable for time (T). The state transition function would take the form $(\Delta X, \Delta Y) = f(a)$, *with* $\Delta T = f(\Delta X, \Delta Y)$. ISE derives paths that are of minimum cost according to an objective function. The objective function accumulates costs over the sequence of actions defining a path, where costs are functions of IPARMS, DPARMS and other parameters.

ISE performs a backwards-chaining search, beginning from one or more goal states. Using a best-first approach directed by the objective function and an admissible heuristic, ISE prioritizes the states to expand. ISE expands a state by backwards-simulating all possible actions from the state. It maintains local constraints by preventing action-state combinations that violate them, and global constraints by monitoring the constrained, path-specific quantities, and propagating only those paths that properly manage them. Each resulting initial state becomes a new node in a directed graph. Every node represents a state from which one of the goals is reachable. Eventually, branches of the graph may intersect the start state IPARMS. Some of these paths may be judged "feasible" according to user-supplied DPARMS criteria. From these, ISE determines the optimal path - the least expensive path according to the user-supplied objective function.

ISE's results are identical to those yielded by A* [19] in an initial plan search with no constraints. However, once environment models have changed locally, ISE operates far more efficiently than A* in re-planning paths. This is a major benefit for TEMPEST, since environment models might evolve as the rover gains new information through sensing. Where changes would force A* to re-build its search from scratch, ISE uses incremental graph theory techniques to repair both the feasible set of solutions and the optimal path within it. The algorithm is time efficient because it determines which portions of the search space are affected by new information and limits the re-computation to those portions. The algorithm is space efficient through the use of three mechanisms:

- Dynamic State Generation: ISE creates a state when it is needed and deletes it when it no longer serves a purpose. This feature precludes the need to allocate an entire multi-dimensional space even though only a small part of it may be searched.
- State Dominance: ISE determines when one state dominates another, through user-defined routines, and prunes the dominated state to minimize unnecessary state proliferation.
- Resolution Pruning: ISE reasons about DPARMS variable resolution and prunes the lesser states from a DPARMS resolution-equivalent class. This feature can dramatically reduce the number of states while still preserving resolution optimality.

ISE enables two modes:

- BESTPCOST finds the minimum cost path. In this mode, the feasible plan with the lowest path cost, as defined by the objective function, is the optimal solution.
- BESTDPARMS finds the "best" state solution below a maximum path cost. In this mode, the objective function serves only to measure path costs against the maximum. The user-defined "better" criterion evaluates DPARMS to prioritize plans that are equal under the objective function to determine the "best" solution.

ISE is a general-purpose, discrete space path search algorithm. To apply ISE to a specific problem, it requires a user to define the domain in terms of the state space S, the actions A, the state transition function $S \times A \rightarrow S$, the start and goal states, and the conditions for feasibility and optimality. The following sections describe how TEMPEST defines these domain elements for long-range navigation.

Domain Models

TEMPEST composes world models and rover models that capture relevant properties of the natural environment, the mission environment and vehicle performance. These models are the foundation for defining the ISE state space, actions and constraints. To ease the computational burden of extending the planning horizon over a day and over one or more kilometers, we have purposely selected coarse models for both the world and rover. They provide reasonable projections of action outcomes under various environmental conditions, but at a resolution that permits reasonable performance on a rover processor. The following subsections illustrate a few examples of world and rover models. Many others are clearly possible. To apply TEMPEST, the list of models must be tailored to the planning problem to encompass all of the desired rover-environment interactions. Table 1 and Table 2 summarize the models currently used in TEMPEST for long-range navigation under solar power.

Table 1: Examples of TEMPEST World Models

Model	Description
Terrain	• Elevation map (DEM) • Slope map • Geodetic reference ellipsoid
Ephemeris	• Solar System body relative position/orientation • Barycentric Dynamical Time reference
Line-of-Sight (LOS)	• Lighting maps, surface-to-surface LOS • LOS maps define instantaneous incident viewing angle on local terrain • LOS map sequences define time-varying surface viewing
Solar Flux	• Estimated peak solar flux

Terrain - The terrain model is founded on a globally-referenced elevation map of the operations area. The terrain model represents generic, raster-patterned digital elevation data. Spatial resolutions for this form of data are typically 10-30 meters per pixel, far larger than the vehicle footprint. The elevation map grid defines the orientation and resolution of the two spatial dimensions of the state space. The software derives slope and slope aspect from the elevation model. Both elevation and slope maps are referenced to standard geodetic biaxial ellipsoids for conversion to latitude/longitude or planet-centered Cartesian coordinates.

Ephemeris - - The CSPICE ephemeris generation software provides relative position and orientation for all major bodies in the Solar System [1]. In counterpoint to the guiding principle of coarse modeling, CSPICE is a very accurate tool, accounting for speed-of-light delays and stellar aberration in determining a body's apparent location. The CSPICE time standard - barycentric dynamical time - is the basis time reference in TEMPEST.

Line-of-Sight Maps - Line-of-sight (LOS) maps encode the elevation angle of a source object above the local ground plane as defined by the slope map (see Figure 2). They also map shadowed locations, where the source is below the ground plane or occluded by other terrain features. LOS maps currently represent incident sunlight for the purpose of modeling solar energy, lighting and shadowing. However, they could also represent line-of-sight to orbiting spacecraft or visibility to fixed points elsewhere on the terrain surface. A ray tracing algorithm projects from the source position (e.g. a Solar System body) onto the terrain model. Sequences of LOS maps, at regular time intervals, represent time-varying visibility.

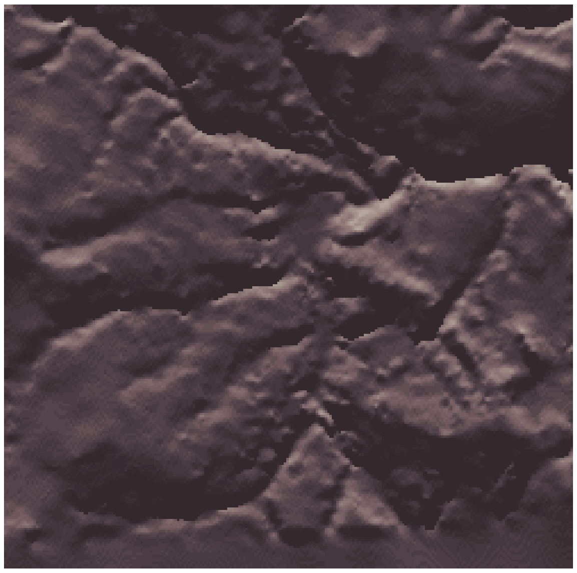


Figure 2: Sample LOS Map for Natural Terrain

Solar Flux - The incident energy per unit area is modeled by the peak flux (as experienced under perpendicular incidence) multiplied by the sine of the elevation angle to capture foreshortening effects. Atmospheric attenuation must be captured in the peak flux value - we currently assume no variation with angle from zenith. We currently ignore all other effects.

Table 2: Examples of TEMPEST Rover Models

Model	Description
Locomotor	• Power load, computed from mass, maximum speed, effective coefficient of friction, drive train efficiency, terrain
Solar Array	• Power source, computed from area, panel efficiency, sun angle of incidence
Nuclear Generator	• Power source, fixed output
Battery	• Energy capacity, from minimum and maximum charge levels
Electronics load	• Power load, fixed value
Navigation cameras	• Constraint management, modeling camera orientation and field-of-view

Rover - This model aggregates rover component models relevant to the long-range navigation planning problem (see Table 2). The rover model enables the computation of state transition costs for actions - for example time and energy. It also models the rover parameters used in constraint checking.

To date, the emphasis of TEMPEST planning has been on energy management. Therefore, most components predict power as a function of activity. Given an action duration, as provided by action target parameters, these components each add or subtract energy from the rover battery. One might envision modeling other resources similarly. For example, mission data, onboard memory and communications with Earth could take the place of energy, battery and solar charging, respectively.

As listed in Table 2, the navigation camera model does not predict power (it would be negligible in comparison with other loads), but specifies a camera orientation and field-of-view. TEMPEST actions all implicitly represent vehicle orientation. The planner uses rover component models with pointing parameters to check constraints, for example to compute sun incidence angle on a solar array or to predict camera sun blinding.

State Space

For the energy management problem, the TEMPEST state space comprises four dimensions - x, y, time and battery energy. The two DEM grid coordinates specify two position IPARMS variables in ISE, X and Y. The CSPICE basis time system, in integer seconds, specifies a third, DPARMS variable T.

The fourth dimension, battery energy E, is more complicated, both in its semantics and how it is represented in ISE. It is important to clarify that E does not represent the instantaneous energy in the battery, but rather the minimum battery energy required to reach the goal. Because ISE searches backwards, from goal to start, it requires a user-specified endpoint goal state (x_g, y_g, t_g, e_g) from which to begin its search. Since there is no operational penalty in arriving at the goal with battery energy greater than e_g, e_g represents the minimum goal arrival energy. With this in mind, let us address how to interpret the effects of positive-energy actions and negative-energy actions.

Actions with net positive energy in the forward-time direction (e.g. solar charging) decrease the value of E in a backwards search. One or more successive "positive energy" actions run backwards could drive E to zero. Herein lies the subtlety: E=0 does not indicate an empty battery, and should not cause ISE to abandon the path instance. Instead, zero E indicates the goal could be reached from the current position and time, even with an empty battery. In tracking backwards, ISE prevents E from dipping below zero - there is no physical meaning to an energy state with "less-than-empty" conditions.

Alternatively, actions with a net negative energy in the forward-time direction (e.g. nighttime locomotion) will tend to cause E to increase in a backwards search. This could cause the value of E to exceed battery capacity. Perhaps counter intuitively, ISE should abandon such a path - it requires a "more-than-full" charge to achieve the goal. Put simply, low E is good, and high E is bad.

A further complication with E is in its representation within ISE. In contrast to position and time, it is not represented as either an IPARMS or DPARMS variable, but within the objective function. We describe how in the *Feasibility and Optimality* section.

Path solutions from ISE are trajectories through this state space. Discrete points in the plans, "waypoints", are 4-tuples of (x, y, t, e).

Actions, Constraints and the State Transition Function

TEMPEST specifies a list of basic motion- and energy-relevant actions that coarsely describe the essential activities of rover navigation. An action is defined by a name, and a target change in IPARMS. Actions that do not affect the IPARMS encode a target change in DPARMS. Given the target parameters, the effect of the action is determined by the specified set of active rover components for the action. Table 3 summarizes the actions and minimum component sets used for recent tests. *Drive*, for example, must incorporate the *Locomotor* component, an electronics load to represent the power from unspecified components, and the navigation cameras to check for sun blinding. A power source of some type (e.g. battery, solar array, nuclear generator) powers the components. Both *Charge* and *Hibernate* are stationary actions, so require a DPARMS target change. Both are fixed-duration actions, but of differing length. However, each can be executed multiple times in series to provide a longer action.

Table 3: Example Action Specifications

Action	IPARMS Target $\Delta X, \Delta Y$	DPARMS Target	Min. Active Components
Drive (8 total)	±1 or 0, ±1 or 0	None	• Locomotor • Electronics load (high) • Navigation cameras • Power source
Charge	0,0	ΔT	• Battery • Electronics load (med./high)
Hibernate	0,0	ΔT	• Battery • Electronics load (low)

A user can also specify local, action-specific constraints that specify state conditions under which an action cannot be executed. Table 4 lists some examples of constraints used in TEMPEST testing. Constraints can be applied in combination to refine the behavior of actions. For example, referring to the actions in Table 3 and constraints in Table 4, one might apply *Maximum Slope*, *Daylight* and *Sun-In-Camera* to the *Drive* actions. This could model a safety limit on slope climbing, and would prevent driving when the cameras cannot detect terrain, either because of darkness or if blinded by the sun. Similarly, *Charge* might apply the *Direct Sun LOS* constraint to distinguish itself from *Hibernate*.

Table 4: Example Local Constraints

Local Constraint	Description
Maximum slope	Prevents actions on slopes above s_{max}
Daylight	Prevents actions during nighttime.
Direct Sun LOS	Prevents actions in shadow or at night.
Sun-In-Camera	Prevents actions where sun vector enters field-of-view of camera.

Given an action and initial state, the TEMPEST state transition function uses a path integrator that calls on the world and rover models to compute the final state.

Specifying Start and Goal States

ISE must know the goal states from which to begin its backwards-chaining search. In the energy management domain, TEMPEST specifies goal positions and battery energies, but leaves arrival time unconstrained. To specify the time variable to ISE, TEMPEST utilizes the ISE multiple goal mechanism. TEMPEST estimates a likely range of arrival times - a "goal window" - based on best-case and worst-case projections on path duration. At even intervals within the goal window, TEMPEST designates separate ISE goal states. All have the same position and energy, but each has a different time value. Since the search graph does not distinguish between paths growing from different goals, path solutions are free to terminate at any interval within the goal window.

TEMPEST also specifies the start state so that ISE can determine which paths are feasible solutions. It designates the nearest grid cell center as the IPARMS position start (x_s, y_s). Since the search occurs on the coarse IPARMS discrete grid, it is reasonable to designate a specific point. However, time is an arbitrarily fine DPARMS variable in the search. A search is not likely to find any path that precisely matches both the start position and a specific time, so TEMPEST designates a "start window" that spans the current time ($t_{si} < t < t_{sf}$). Finally, TEMPEST designates the current battery energy e_s as the maximum energy for the start of any feasible plan.

Feasibility and Optimality

Using this specification of the start and goal states, ISE searches for feasible plans from which to select an optimal plan. Candidate plans are sequences of states, or waypoints:

$$P = \{w_1, \ldots, w_n\} \text{ with } w_i = (x_i, y_i, t_i, e_i),\ t_i < t_{i+1} \quad (1)$$

Given a start state $s = (x_s, y_s, t_s, e_s)$ and a start window time interval $[t_{si}, t_{sf}]$, a plan ***P*** is feasible if and only if:

$$(x_1 = x_s) \wedge (y_1 = y_s) \wedge (t_{si} \le t_1 \le t_{sf}) \wedge (e_1 \le e_s) \qquad (2)$$

In both the Sun-Synchronous Navigation project [31] and Life in the Atacama project [38][32], TEMPEST sought energy-optimal paths. Unlike path duration or distance, whose quantities increase monotonically as a plan gets longer, energy is non-monotonic - locomotion and other rover activities expend energy while solar energy and other power sources restore it. Standard heuristic search approaches to path planning would become "stuck" in states providing a net positive energy intake, and would never actually reach the designated goal.

TEMPEST handled this problem differently for the Sun-Synchronous Navigation and Life in the Atacama projects. In the Arctic, TEMPEST used ISE in its BESTDPARMS mode with path duration as the objective function. By setting a maximum duration cost on plans, TEMPEST discarded partial plans that were "stuck" in energy-rich states. Furthermore, during the search ISE kept or discarded paths according to a "better" criterion: if two graph branches independently reached the same position and time with the same duration cost, ISE would remove the branch requiring more energy to reach a goal. Solutions were "energy-optimal" over their entire length, but under the restriction of the maximum duration.

For Life in the Atacama, TEMPEST used ISE in its BESTPCOST mode and a new objective function that also yielded energy-optimal paths, but with vastly improved performance. ISE enables a composite objective function consisting of two or more elements that can be tracked and manipulated separately, but that collectively contribute to a single objective function cost. To describe the specific composite objective function used in the Atacama experiments, we define the quantity $E_{\max}$:

$$E_{max} = \left| min_{\forall (s \in S,\, a \in A,\, \Delta e \le 0)} \Delta e = f(s, a) \right| \qquad (3)$$

In words, $E_{\max}$ is the absolute value of the greatest single-step negative cost (positive increment) to battery energy over all states and actions, for example through solar charging. The value of the TEMPEST objective function is the sum of two quantities:

$$L = nE_{max} \qquad (4)$$

$$B = \sum_{i=1}^{n} \Delta e_i \qquad (5)$$

L is a measure of plan length in increments of E_{max}, and B is the sum of energy costs (positive and negative) over the path. At each plan step, the objective function changes by $\Delta e + E_{max} \ge 0$. At the most energy-rich state and action the components cancel. For all others, the increment is positive. The objective function increases monotonically over a path. Using this objective function under the BESTPCOST mode, ISE produces paths that are optimal in combined terms of minimum plan length and energy cost.

The B term in the objective function can be used to track the E state variable. Further, since the ***X***, ***Y*** and ***T*** variables are totally independent of ***E*** in all TEMPEST applications to date, ***E*** can be removed from the search space. Collapsing the four-dimensional search to three dimensions drastically reduces the computation and memory for search, thereby enabling TEMPEST to run online and to solve larger planning problems.

Single-Goal Planning

The simplest TEMPEST task is to plan from a start state to a single goal position. The previous sections describe the essential elements of the process. Given a start state, a goal position and a goal battery energy, the software computes a window of goal states, then calls ISE to search from all goals. Under the BESTPCOST mode, the first detected feasible path will also be the optimal. Under BESTDPARMS mode, ISE finds the "best" feasible state that falls below the path cost maximum. The resulting path begins at the start state, within the start window, and arrives at one of the goals in the goal window.

Multiple-Goal Planning

By chaining ISE searches in series, TEMPEST enables planning to an ordered list of position goals ***G*** $= \{g_1,\ldots,g_n\}$, $g_i = (x_i, y_i)$. TEMPEST creates a separate ISE graph for each path "segment" between goals. As with ISE search, planning for multiple goals happens in reverse order, from the last segment to first segment. Multiple goal planning is similar to single goal planning, but differs in some important ways.

Most importantly, the optimal solution for an arbitrary segment is not necessarily part of the optimal solution for the entire goal list. Therefore, it is incorrect to simply chain locally-optimal segments to form the globally-optimal solution. Instead, TEMPEST tracks multiple path candidates through all the segments, then solves for the optimal surviving candidate in the first segment. We describe the process below.

Figure 3 depicts the algorithm for multiple-goal planning. The horizontal axis is time, and the vertical axis, goal positions, from the start at the bottom to the final goal g_n at the top. Note that the time axis may span one or more days, as denoted by the light and shaded regions corresponding to "noon" and "midnight". Each vertical interval between goals represents one of the "segments" of the total path. As with single-goal planning, the process for segment planning proceeds in backwards-chaining order, from the top right of the diagram to the bottom left.

We begin by defining the start and goal states. TEMPEST defines the start state as in single-goal planning. The diagram in Figure 3 depicts this state and the initial "start window" as a blue square on the left of the horizontal axis. For goal states, as with single-goal planning, TEMPEST projects the best- and worst-case durations for each segment, as depicted by the red dashed diagonal "bounding lines" ascending up and right from the initial start window. The left, steep bounding line represents the fastest possible travel for each segment. The right bounding line is less steep, taking into account possible stationary actions and worst-case path length. The intersection of the bounding lines with the horizontal goal lines defines the time range for both the start window of one segment, and the goal window of the previous segment. The interval at the final goal g_n defines the final goal window. As in single-goal planning, the individual goal states are defined at even intervals over the goal window (see *Specifying Start and Goal States* above). Figure 3 shows these as a horizontal array of blue squares at the goal g_n.

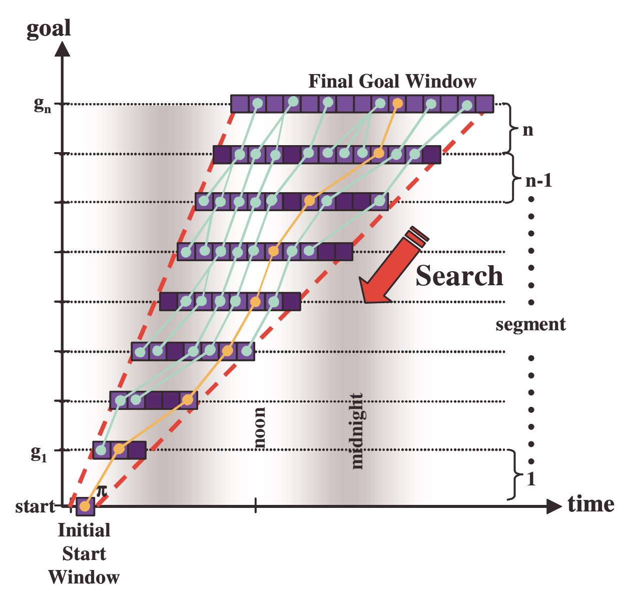


Figure 3: Diagram of Multi-Goal Planning

To plan the final segment, TEMPEST defines a start window using the same bounding lines as for the goal. It defines start times at even intervals in this window, and sets the start energy to full battery capacity. This leaves the energy unconstrained - according to feasibility in equation (2), any solution that arrives at g_{n-1} below the start energy satisfies the energy criterion.

Recall that the locally optimal segment solution is not, in general, part of the global optimal. To generate plan candidates for the segment, TEMPEST runs ISE separately for each interval in the start window. Each search results in an optimal solution for the particular start state interval, assuming feasible paths exist. Figure 3 depicts these solutions as cyan segments descending to the next lowest goal[2]. The set of all candidate paths is the solution for the segment.

To plan the next earliest segment, TEMPEST defines goal states equal in value to the start states of the optimal paths from the previous segment. The effect is to chain segment solutions together to build long, consistent plans one segment at a time. The process repeats for all the segments. The process is the same for the initial segment, except that there is only one start state. The initial segment yields a single path. The chain of segment plans, beginning with the initial segment plan, is the global optimum for the goal list, depicted by the orange segment chain in Figure 3.

2. Note that some goal states and start states do not produce feasible solutions. Further note that each start state can only have one goal state (the optimal path is unique, barring ties), but that the optimal paths from several start states may all arrive at the same goal state.

Plan Extension

Invariably, mission execution does not follow plans precisely. Often, unforeseen operational events cause deviations from the route, schedule or energy guidelines. In response to these deviations, TEMPEST updates plans by re-running ISE with the most current initial rover state. ISE simply extends its graph to the new state, yielding a new optimal plan. There is no guarantee that the updated solution is similar to the original. This is the most basic form of re-planning.

Re-Planning

Models of the world and rover may evolve over time as new data is gathered. A planner that enables re-planning to can be far more independent from Earth-bound control - it adapts to changes and continues operations autonomously. Through ISE, TEMPEST enables highly efficient re-planning.

Re-planning must occur when changes in the world or rover models alters the cost for actions. Given model updates, TEMPEST reports the changes for each affected IPARMS state (the change may also be DPARMS dependent). ISE determines the states affected by the change and, in subsequent calls, repairs the nodes in the graph to reflect the new optimum. It is efficient because it restricts its computations to the set of nodes affected by the changes.

Typically, new data about the world comes from rover sensors. Under the backwards-chaining search, the rover position is at the leaves of the search graph. Therefore, changes deriving from rover sensor data affect only the *ends* of the graph, and are inexpensive. In contrast, global world model changes and changes to rover parameters often affect a large portion of the search graph. Local re-planning typically requires less than 1% of the time for an initial search. The more global the scope of changes, particularly in the area near the goals, the more re-planning mimics the computational cost of initial path search.

TEMPEST also provides re-planning for multi-goal traverses. This contrasts with single-goal re-planning in that model changes may affect more than one segment, and hence more than one ISE graph. TEMPEST must notify each affected ISE instance, but need only initiate re-planning from the latest affected segment. As with single-goal re-planning, rover-local model updates are the least computationally expensive.

Now that we've described how TEMPEST and ISE work, we briefly discuss how TEMPEST might be used for rover exploration.

5. Operational Concept

Placing TEMPEST into a rover architectural framework is a bit of a dilemma. Path planning software is often categorized as low-level control software, as in robotic manipulator systems. It also re-plans at reactive time scales, reinforcing the low-level classification. However, TEMPEST defines coarse action sequences for day-long, multikilometer traverses, and defines when and where battery charging, hibernation and other activities occur. These tasks are typically reserved for high-level planners.

Furthermore, TEMPEST works at very coarse resolutions. It treats terrain traverse abstractly, ignoring features below the scale of the elevation model. It has no detailed knowledge of rover component interactions or activity scheduling to achieve a task. It plans only sequential actions. So, it is neither sufficiently capable to serve as the only path planner nor is it meant to serve as the only planner-scheduler for a robot.

We view TEMPEST and planner-scheduler software as complementary components in a high-level planning system. A planner-scheduler and TEMPEST might negotiate to arrive at a mutually acceptable plan (see Figure 4). The planner-scheduler would receive mission specifications from human operators, and decompose the mission goals into smaller tasks. It might determine when extensive travel was necessary, and call upon TEMPEST to provide plans that meet goal conditions and satisfy constraints for those mission phases. The plans might force the planner-scheduler to re-sequence other portions of its plan to accommodate travel requirements as imposed by TEMPEST.

In very simple rover missions, as demonstrated by our team, TEMPEST could serve as the only planner. Finally, though designed principally as an online process, TEMPEST could be split into onboard and offboard elements. The offboard element could provide initial planning, then transmit the ISE graph to the rover to allow subsequent re-planning.

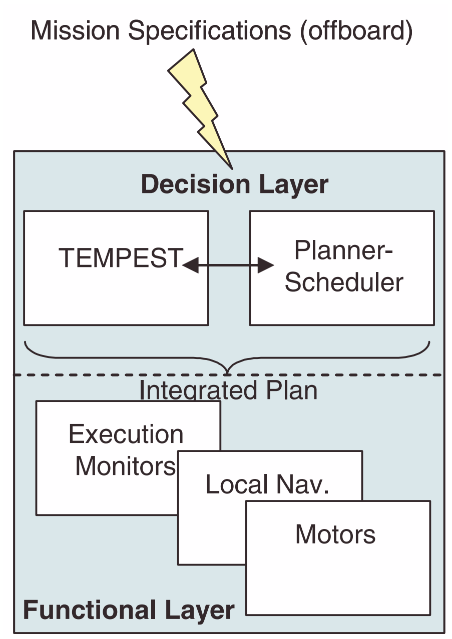


Figure 4: One Model of TEMPEST's Role in a Rover Architecture: TEMPEST and a Planner-Scheduler negotiate to derive plans that are mutually agreeable.

In terms of navigation, we envision TEMPEST at the top of a layered hierarchy. The distance between TEMPEST waypoints depends on the spatial resolution of its basis elevation map grid, typically 10-30 meters. This distance is commensurate with the goal distances for existing local navigation software [8][13][25]. Where TEMPEST is ignorant of local, small-scale terrain, local navigation planners combine obstacle detection with path planning. Each TEMPEST waypoint would serve as a global goal for the local navigator.

6. Preliminary Results

Simple Example

We present a sample planning problem to illustrate TEMPEST behaviors. A contour map in Figure 5 shows the elevation profile for synthesized terrain on a mock Martian surface. Mountains form a central pattern of valleys running in a North-South (up-down) direction, and a rounded crater lies to the northeast. The rover starts in the morning at the southeast corner of the map ("Start"), and must traverse to the northwest corner ("Goal 2"). Scientists designate a target for investigation in the valley, "Goal 1", en route to the final destination. Mission engineers (or an onboard planner-scheduler) require the robot to reach Goal 2 with 100 W-hr of charge left for subsequent operations.

Unfortunately, the elevation map provided to the rover is incorrect. Its preliminary map indicates a clear exit from the valley system at its northern extreme, between two peaks. The actual terrain involves a far higher and steeper pass, beyond the locomotion capability of the rover.

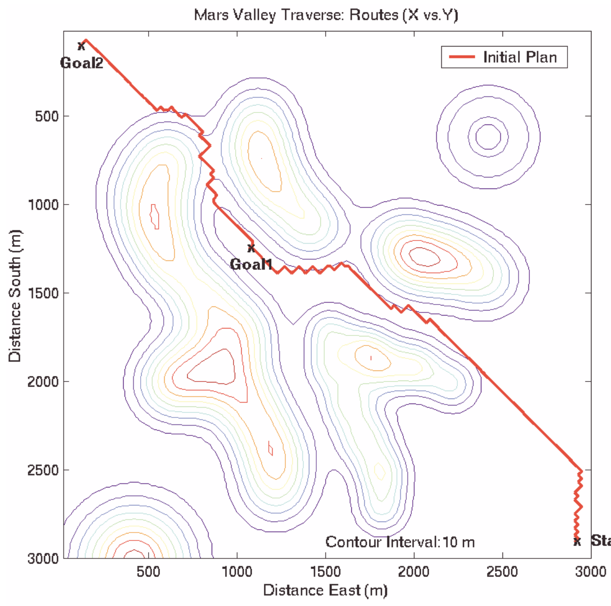


Figure 5: Initial Plan Route: TEMPEST plans a path through the valley exit to the northwest.

In this example, we run a simple simulation of the mission. At each step, the simulated rover plans a path from its current state, then executes one step of the plan. At each new point, the rover "senses" the local environment, detecting the actual elevation, slope and lighting of all cells within two pixels. Based on this new data, TEMPEST re-plans a path and the process continues.

Figure 5 shows the *initial* TEMPEST plan route. With the exception of a few minor path deviations, the route follows a direct path from the start through each of the goals. Subsequent re-plans during "execution" yield similar routes. The solid red curve in Figure 6 shows the timing for the initial plan. The slope of the curve represents the rover speed toward Goal 2. One observes that it is only slightly slower than the theoretical fastest, straight-line approach, as shown by the steep dash-dot line. The red solid line in Figure 7 shows the required battery energy profile for the initial plan. The plan allows the robot to begin with an empty battery, and only requires increasing charge at the end of the plan to meet the Goal 2 requirement. This indicates that solar energy provides ample energy for locomotion.

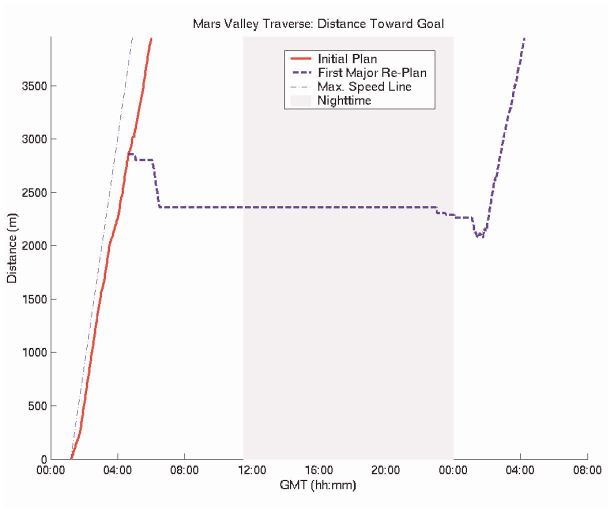


Figure 6: Distance Towards Goal 2 vs. Time: The initial plan follows a very direct path. The detour requires that the rover endure a night in hibernation, as shown by the flat region of the re-plan curve.

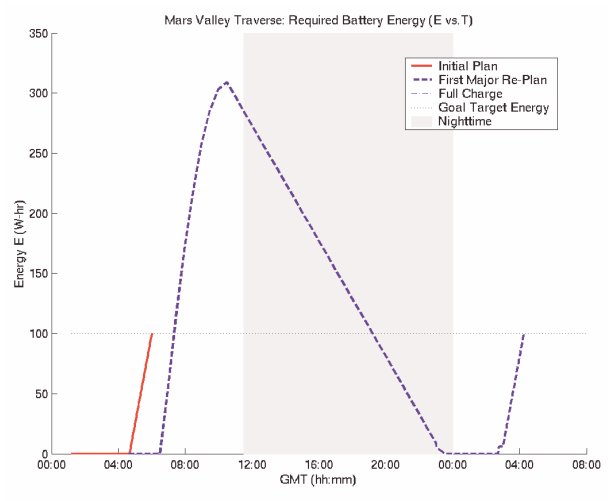


Figure 7: Battery Energy Requirement vs. Time: The initial plan can start from total discharge to reach the goal charge. The detour requires the rover to perform stationary charging to nearly full capacity in order to endure the nighttime.

The simulated rover reaches Goal 1 and continues without stopping toward Goal 2. The nearest valley exit, according to its internal elevation map, lies to the northwest directly in line with its next goal. However, as it approaches the supposed low pass, the robot detects the steep, intraversible pass. Figure 8 shows the first substantial re-plan in the sequence, based on this discovery. With no escape to the northwest, TEMPEST selects the least expensive alternative - a detour through a narrow valley to the northeast (the blue dashed line). This new route is a significant departure from the original. The extra distance means that the robot cannot reach Goal 2 before sundown.

The original plan did not anticipate the extra burden of nightfall on battery reserves. However, TEMPEST determines a prolonged *Charge* action followed by overnight *Hibernation* will enable it to reach Goal 2 by late morning the following day. Figure 6 shows the rate of travel towards Goal 2 for the detour as a dashed blue line. Note that the robot must first reverse course, and then remains stationary for nearly 18 hours, first in sunlight (charging batteries), then overnight (hibernating, in the shaded region). The following morning, the rover continues its course around the mountain, then moves to Goal 2.

Figure 7 shows the required battery energy profile over the same time span, also with a blue dashed line. Note that the re-plan still enables the robot to start from an empty battery. However, well in advance of nightfall, the plan requires a steady increase in battery charge to generate reserves for the night. The battery energy requirement falls overnight - the morning sun is sufficient to charge the battery to the required Goal 2 level.

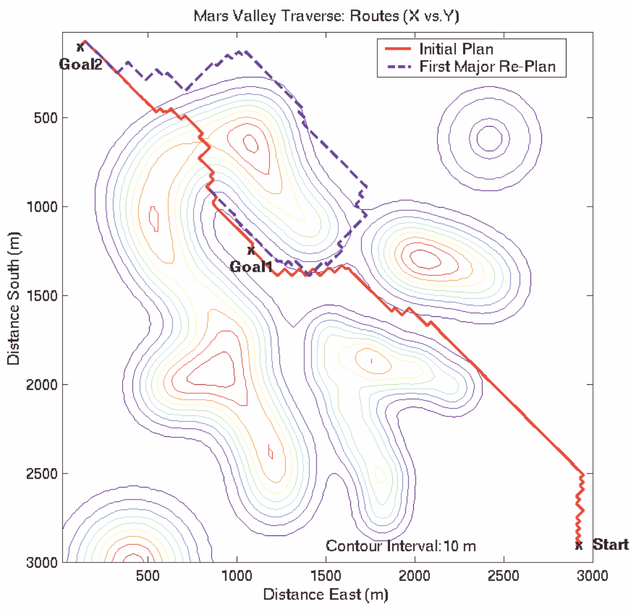


Figure 8: First Significant Re-Plan Route: The robot discovers a much steeper, taller pass at the end of valley, prompting a detour around the mountain.

This example highlights how TEMPEST coordinates route, timing and battery energy to achieve multiple goals. Unlike many approaches to temporal planning, TEMPEST approaches the problem as a whole rather than by simpler, but sub-optimal, hierarchical breakdown. Incorporating this capability into a rover could provide a significant boost to rover safety, mission time efficiency and reliability.

Life in the Atacama Project

In April 2003, TEMPEST was deployed as the long-range path planner for the solar-powered Hyperion robot as part of the first year of field experiments for the Life in the Atacama (LITA) project [38]. We briefly summarize some of our findings in this paper. For a more complete account of TEMPEST results, please refer to [32] and [38].

LITA operated in Mars-relevant terrain in the Atacama Desert in Chile. In support of the development of technologies for robotic astrobiology, the first year's field experiments focused on achieving basic autonomy and reliable long-distance travel. TEMPEST was the sole high-level planner for Hyperion (Figure 9) during this first, navigation-dominated year. There were seven days of fully integrated navigation experiments, during which TEMPEST generated 27 plans and 83 plan extension re-plans. Of those, several resulted in single-command traverses of over 200 meters; the longest fully autonomous execution covered over 1100 meters.

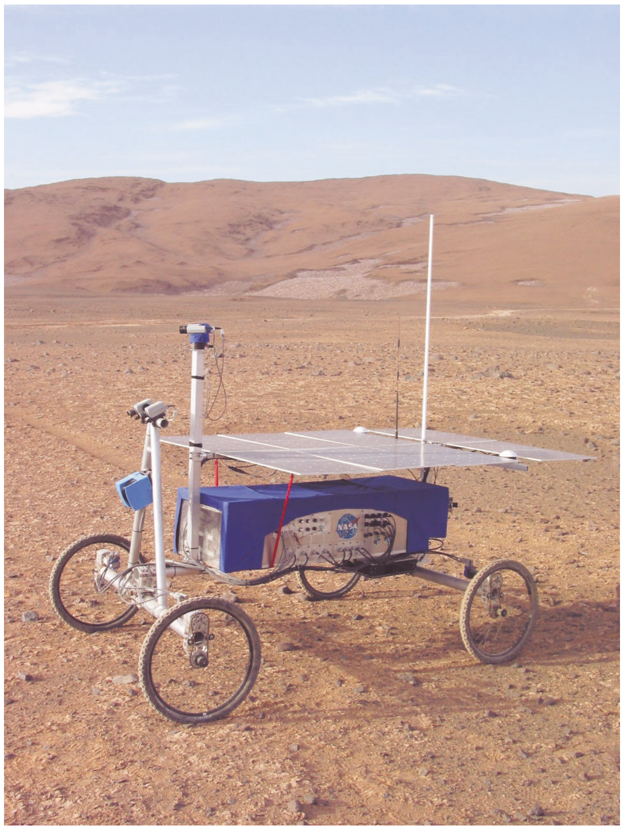


Figure 9: The Hyperion Robot: TEMPEST provided online navigation planning and re-planning for several days of Mars-relevant experiments.

During rover operations, a simple executive process managed operator plan requests, coordinated plan execution, and triggered re-planning. Human operators transmitted single latitude/longitude position goals to the rover. The executive passed a plan request to TEMPEST for the goal, and recorded the solution. The executive executed each action in sequence, with calls to the local navigator for Drive actions, and by pausing for Charge actions. A health monitor process monitored execution timing. If Hyperion progress was too slow relative to the plan, the monitor triggered a fault, prompting the executive to suspend the current execution and to request a re-plan. Figure 10 shows a detailed view of how successive plans (shown as thin, dotted traces with point markers) varied with the evolving rover state.

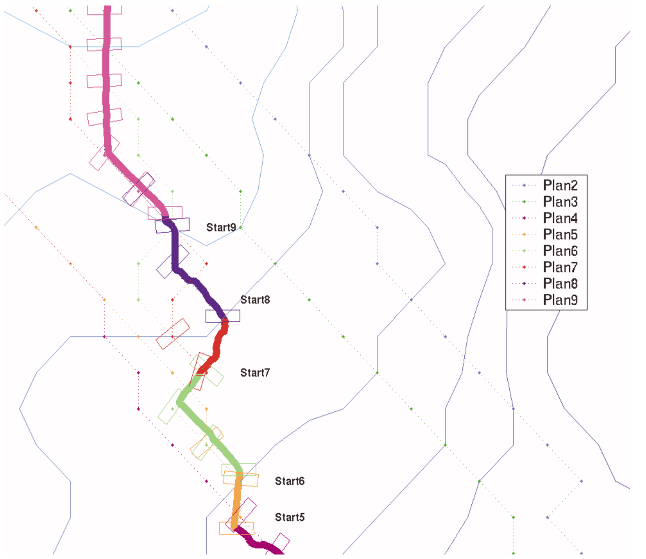


Figure 10: Planned and Executed Paths: Plans progress from bottom to top. Goal regions surrounding TEMPEST plan waypoints acted as global goals for a local navigator. Contours express the elevation profile.

The two-layer navigation hierarchy resulted in accurate and operationally smooth driving. TEMPEST provided waypoints, centered at 30 meter grid intervals, as global goals for the local planner. To avoid placing goals in the center of local obstacles, TEMPEST provided the local navigator with rectangular goal regions (see Figure 10). The local navigator used stereo vision to detect hazardous rocks, and a planner that yielded safe paths to goal regions.

Computational and Memory Performance

TEMPEST operation can be divided into three phases: Pre-Calculation, Initial Planning, and Re-Planning. The Pre-Calculation phase computes slopes from the elevation map, then pre-computes transition costs for the entire state space and action space. Initial Planning is the most expensive search, when ISE builds the graphs and yields an initial solution. Re-Planning happens from that point as execution continues.

The Pre-Calculation phase is dominated by transition cost computation. Both computation and memory are $O(|S||A|)$ complexity, where S is the state space and A is the action space.

Quantifying the Initial Planning phase is not straightforward. The complexity is related to the search branching factor and depth. Exhaustive expansion of actions causes a branching factor of $|A|$, but the ISE dynamic state, state dominance and resolution pruning mechanisms effectively reduce this. The degree to which they help is highly domain dependent. The search depth also depends on the domain. Searching over homogeneous, obstacle-free terrain would result in a shallower search than a search where the optimal path is circuitous.

Re-Planning phase is also difficult to quantify. Re-Planning phase complexity is lower than that of Initial Planning, because ISE limits graph repair to the portion of the graph affected by changes in transition costs. The closer the changes occur to the goal state, the more re-planning matches the computation cost of initial planning.

With this said, we provide a single example of TEMPEST performance, in a simulation context, to provide a feel for computational and memory demands. Figure 11 depicts the memory allocation for TEMPEST over all three phases as a function of time for a typical planning problem taken from the LITA field experiment. The experiment ran on a Pentium 1.2 GHz machine with 1 GB of RAM.

In the Pre-Calculation phase, TEMPEST computed the slope for the entire 2527-by-2356 pixel DEM (shown in Figure 11 by the flat memory allocation in the first 88 seconds), and costs for 10 actions for each of 5.1 million states (shown as a gradually increasing slope of memory allocation ending at 640 seconds elapsed time). Though expensive, the Initialization phase nominally happens only once for a given set of goals.

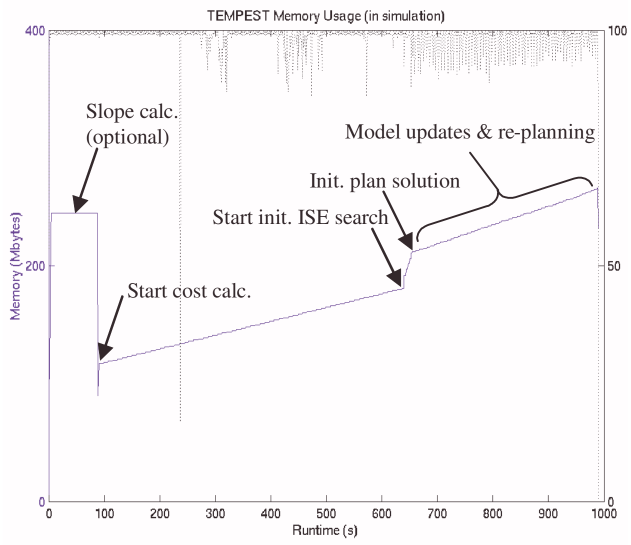


Figure 11: Computational and Memory Performance: State transition cost pre-calculation and initial search are expensive but infrequent. Re-planning is inexpensive, but more common.

The Initial Planning phase begins immediately after Pre-Calculation. In this scenario, ISE searches over a 75-by-24 pixel area, and finds an optimal plan containing 64 waypoints.

During actual rover operations, re-planning can occur at any time after TEMPEST generates an initial plan. This simulation follows the Initial Planning phase immediately with 63 back-to-back action simulations and re-plans. After each simulated action, the simulator alters the terrain model properties in a two-cell radius from the rover to model rover sensing. TEMPEST re-plans at each waypoint to re-optimize the path considering the newly updated model. Re-planning.

7. Conclusions

Rover autonomy promises to significantly improve science data return and safety while reducing operations costs for planetary exploration missions. TEMPEST addresses a new level of navigational autonomy that plans routes, sequences activities, manages resources, and enforces constraints. Working in conjunction with Planner-Scheduler software, TEMPEST could add a solid grounding for segments of a mission in which travel is a primary activity.

TEMPEST enables planning in a state space of position, time and battery energy, and enables highly efficient re-planning in response to updates in world and rover models. Experiments in simulation show that the planner derives plans that balance the considerations of route, timing and energy, even when presented with unexpected situations. Field experiments on an actual rover demonstrate that TEMPEST selects reasonable routes through large-scale terrain, and adapts well to operational delays through policy extension.

The planner has several limitations - these may or may not be problematic for future planetary missions. TEMPEST does not represent parallel actions nor partial-ordering as many generic planners do. However, we expect that it would operate in conjunction with a Planner-Scheduler, and could provide sufficiently detailed action modeling to side step the issue.

Perhaps more importantly, TEMPEST does not explicitly address uncertainty. The planetary exploration domain is rife with uncertainty - rover behavior, unknown terrain, operational variations to name a few sources. Re-planning is a “last-minute” approach to uncertainty. In many situations, re-planning might prove to be “too little, too late.” Optimal plans often pass very close to hazardous situations. This proximity means that small delays or uncertain state estimation might cause a robot to pass into danger.

By committing to an objective function relating to one aspect of rover operations, TEMPEST may not be able to perform well in all situations. TEMPEST currently uses a combined metric of energy and plan length to optimize. However, even for a an application to solar or RTG-powered operations, this may not always be the right metric. By specifying a more general metric, grounded in achieving mission objectives, might be able to address a wider range of scenarios. Alternatively, a rover might be able to create different instantiations of TEMPEST, depending on the specific behavior mission operations wish the robot to exhibit.

8. Current and Future Work

Implementation

In connection with the Mars Technology Program and CLARAty [34] development, our team is currently re-implementing TEMPEST to comply with the CLARAty coding standards, and to significantly improve TEMPEST functionality. The new implementation is written in C++, and leverages substantially from the object-oriented paradigm. At the lowest level, the new design will use CLARAty base classes (e.g. vectors, arrays, matrices). More importantly, C++ classes will replace the procedural code that dominates the current implementation.

The revised implementation will enable a much richer specification of planning problems, and will require far less effort to re-adapt the software to different scenarios, or to replace or augment the current search algorithm. The modified implementation will also allow TEMPEST to pull rover models from the CLARAty functional layer, thereby avoiding proliferation of models that plagues some software systems.

Planning Under Uncertainty

Both the Sun-Synchronous Navigation project and the first year of the Life in the Atacama project highlighted vulnerabilities of the TEMPEST approach to uncertainty. We are currently pursuing techniques for efficiently (and approximately) planning under the most common sources of uncertainty. This includes imposing constraints defined by confidence intervals on arrival time and battery energy within ISE, to employing a Markov Decision Process to account for various control noise in plan execution. Our hope is that future versions of TEMPEST will guard against some forms of uncertainty, thereby improving rover safety margins.

High-Fidelity Simulation Testing

Given that extended field experiments are rare and limited in duration, it is our goal to test TEMPEST more exhaustively under high-fidelity simulation. We intend to test on ROAMS, an advanced rover simulator developed at NASA JPL [39]. To generate statistics on the sensitivity of TEMPEST plans to uncertainty, we intend to utilize synthesized terrain [7] and introduce noise to the basis simulation models, then have TEMPEST repeatedly plan and re-plan routes during simulated traverses. To date, all our data regarding uncertainty is anecdotal, and isolated. By conducting repeated realistic simulations, it is our hope to generate a more concrete basis for developing planning under uncertainty, as well as for general development.

9. Acknowledgments

This research was funded under the Mars Technology Program, the NASA Life in the Atacama project, and the NASA Sun-Synchronous Navigation project. We wish to thank all the participants of those projects for their support in designing, developing and testing TEMPEST.

References

[1] C. H. Acton Jr., "Ancillary Data Services of NASA's Navigation and Ancillary Information Facility", *Planetary and Space Science*, 44 (1):65-70, 1996.

[2] D. Bernard, G. Dorais, E. Gamble, B. Kanefsky, J. Kurien, G. Man, W. Millar, N. Muscettola, P. Nayak, K. Rajan, N. Rouquette, B. Smith, W. Taylor, Y. Tung, "Spacecraft Autonomy Flight Experience: The DS1 Remote Agent Experiment," *Proceedings of the AIAA Conference 1999*, Albuquerque, NM.

[3] J. Bobrow, S. Dubowsky, J. Gibson, "Time-Optimal Control of Robotic Manipulators Along Specified Paths," *International Journal of Robotics Research, Vol. 4, No. 3*, Fall 1985.

[4] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, D. Tran, "ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling," *SpaceOps 2000*, Toulouse, France, June 2000.

[5] J. Cutts, S. Hayati, D. Rapp, C. Chu, J. Parrish, D. Lavery, R. DePaula, "The Mars Technology Program," *Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics & Automation in Space (i-SAIRAS 2001)*, St-Hubert, Quebec, Canada, June, 2001.

[6] T. Fraichard, "Dynamic Trajectory Planning with Dynamic Constraints: a 'State-Time Space' Approach," *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 93)*, Yokohama, Japan, July 1993.

[7] R. Gaskell, J. Collier, L. Husman, R. Chen, "Synthetic Environments for Simulated Missions," *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, March 2001.

[8] S. Goldberg, M. Maimone, L. Matthies, "Stereo Vision and Rover Navigation Software for Planetary Exploration," *Proceedings of the 2002 IEEE Aerospace Conference*, Big Sky, MT, March 2002.

[9] A. Howard, B. Werger, H. Seraji, "Integrating Terrain Maps into a Reactive Navigation Strategy," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2003)*, Taipei, Taiwan, September 2003.

[10] K. Iagnemma, H. Shibly, A. Rzepniewski, S. Dubowsky, "Planning and Control Algorithms for Enhanced Rough-Terrain Rover Mobility," *Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2001),* St-Hubert, Quebec, Canada, June 2001.

[11] L. Kavraki, P. Svestka, J-C Latombe, M. Overmars, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE Transactions on Robotics and Automation*, Vol. 12, No. 4, August 1996.

[12] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," International Journal of Robotics Research, Vol 5. No. 1, p 60, 1986.

[13] S. Laubach and J. Burdick, "RoverBug: An Autonomous Path-Planner for Planetary Microrovers," *Sixth International Symposium on Experimental Robotics (ISER 99)*, Sydney, Australia, March 1999.

[14] S. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," *TR 98-11, Computer Science Department, Iowa State University*, October, 1998.

[15] B. Logan, N. Alechina, "A* with Bounded Costs," *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison, WI, July, 1998.

[16] N. Nilsson, "A Mobile Automation: an Application of Artificial Intelligence," *Proceedings of the Interna-*

tional Joint Conferences on Artificial Intelligence, 1969.

[17] C. O'Dunlaing, M. Sharir, C. Yap, "Retraction: A New Approach to Motion Planning," *ACM Symposium on Theory of Computing,* 15:207-220, 1983.

[18] D. Pai, L. Reissell, "Multiresolution Rough Terrain Motion Planning," *IEEE Transactions on Robotics and Automation, Vol. 14, No. 1*, February 1998.

[19] J. Pearl, Heuristics: Intelligent Search Strategies for Computer Problem Solving, Addison-Wesley, Reading, MA, 1984.

[20] R. Richbourg, N. Rowe, M. Zyda, R. McGhee, "Solving Global Two-Dimensional Routing Problems using Snell's Law and A* Search," *Proceedings of the 1987 IEEE International Conference on Robotics and Automation (ICRA 87)*, Raleigh, NC, March 1987.

[21] N. Rowe, "Roads, Rivers and Obstacles: Optimal Two-Dimensional Path Planning around Linear Features for a Mobile Agent," *International Journal of Robotics Research 9*, no. 6, pp. 67-74, December 1990.

[22] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice-Hall Publishers, Upper Saddle River, NJ, Copyright 1995.

[23] K. Shillcut, "Solar Based Navigation for Robotic Explorers," Ph.D. thesis, technical report CMU-RI-TR-00-25, October 2000.

[24] Z. Shiller, Y. Gwo, "Dynamic Motion Planning of Autonomous Vehicles," *IEEE Transactions on Robotics and Automation, Vol. 7, No. 2*, April 1991.

[25] S. Singh, R. Simmons, T. Smith, A. Stentz, V. Verma, A. Yahja, K. Schwehr, "Recent Progress in Local and Global Traversability for Planetary Rovers", *Proceedings from the 1999 IEEE International Conference on Robotics and Automation (ICRA 99)*, 1999.

[26] A. Stentz, "The Focussed D* Algorithm for Real-Time Replanning", *Proceedings of The 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada, August 1995.

[27] A. Stentz, M. Hebert, "A Complete Navigation System for Goal Acquisition in Unknown Environments", *Autonomous Robots, Vol. 2, No. 2*, August 1995.

[28] A. Stentz, "Optimal and Efficient Path Planning for Unknown and Dynamic Environments", *International Journal of Robotics and Automation*, Vol. 10, No. 3, 1995.

[29] A. Stentz, "CD*: A Real-time Resolution Optimal Re-Planner for Globally Constrained Problems," *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, Edmonton, AB, Canada, July 2002.

[30] A. Stentz, "Optimal Incremental Search for High-Dimensional, Constrained Path Finding Problems," *Carnegie Mellon Robotics Institute Technical Report*, to be released 2002.

[31] P. Tompkins, A. Stentz, W. Whittaker, "Mission Planning for the Sun-Synchronous Navigation Field Experiment," *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA 02)*, Washington D.C., May 2002.

[32] P. Tompkins, A. Stentz, W. Whittaker, "Experiments in Mission-Level Path Execution and Re-Planning," submitted to the *Conference on Intelligent Autonomous Systems (IAS 04)*, Amsterdam, Netherlands, 2004.

[33] C. Urmson, "Locally Randomized Kinodynamic Motion Planning for Robots in Extreme Terrain," Ph.D. thesis proposal, The Robotics Institute, Carnegie Mellon University, 2002.

[34] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, H. Das, "The CLARAty Architecture for Robotic Autonomy," *Proceedings of the 2001 IEEE Aerospace Conference*, Big Sky, MT, March 2001.

[35] R. Volpe, S. Peters, "Rover Technology Development and Infusion for the 2009 Mars Science Laboratory Mission," *Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2003)*, Nara, Japan, May 2003.

[36] D. Wettergreen, B. Shamah, P. Tompkins, R. Whittaker, "Robotic Planetary Exploration by Sun-Synchronous Navigation", *Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 01)*, Montreal, Canada, 2001.

[37] D. Wettergreen, M. B. Dias, B. Shamah, J. Teza, P. Tompkins, C. Urmson, M. Wagner, W. Whittaker, "Experiments in Sun-Synchronous Navigation," *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA 02)*, Washington D.C., May 2002.

[38] D. Wettergreen, N. Cabrol, F. Calderon, M. Deans, D. Jonak, A. Luders, F. Shaw, T. Smith, J. Teza, P. Tompkins, C. Urmson, V. Verma, A. Waggoner, M. Wagner, "Life in the Atacama: Field Season 2003 Experiment Plans and Technical Results," *CMU technical report CMU-RI TR-03-50*, October, 2003.

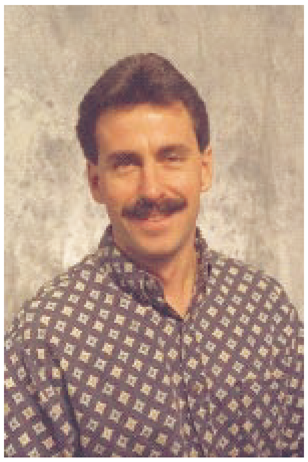
[39] J. Yen, A. Jain, "ROAMS: Rover Analysis Modeling and Simulation Software," in *Proceedings of the 5th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 99)*, Noordwijk, Netherlands, June 1999.

Author Biographies

Paul Tompkins is a Doctoral Candidate in The Robotics Institute at Carnegie Mellon University. He received his M.S. in Robotics from Carnegie Mellon University in 2001, his M.S. in Mechanical Engineering from Stanford University in 1997, and his B.S. in Aeronautics and Astronautics from MIT in 1992.

Following his undergraduate program, he worked for five years at Hughes Space and Communications Company as a mission analyst and orbital operations team member. During this time he was a lead mission analyst for several geostationary commercial satellite programs, and also led several orbital operations campaigns from liftoff to on-orbit delivery.

Mr. Tompkins returned to full-time academic status in 1998 at Carnegie Mellon University, where he researches navigation planning for planetary rover applications. He was a participant in the Sun-Synchronous Navigation project, during which he initiated work on TEMPEST. The project culminated in a field experiment on Devon Island in the Canadian Arctic in July 2001. He remains lead developer for TEMPEST in support of the Life in the Atacama project and under the Mars Technology Program. He is the recipient of the Hughes Electronics Graduate Fellowship and is currently a NASA Graduate Student Research Program Fellow, sponsored under NASA Ames Research Center.



Dr. Anthony Stentz is a Research Professor at the Robotics Institute, Carnegie Mellon University, and Associate Director of the Robotics Institutes National Robotics Engineering Consortium. He received his Ph.D. in computer science from Carnegie Mellon University in 1989, his M.S. in computer science from CMU in 1984, and his B.S. in physics from Xavier University of Ohio in 1982.

Dr. Stentz's research expertise includes unmanned ground vehicles, unmanned air vehicles, dynamic planning, multi-vehicle planning and coordination, perception for mobile vehicles, robot architecture, and artificial intelligence in the context of field worthy systems. Dr. Stentz was the first to merge the fields of incremental algorithms and heuristic search to produce very fast re-planners for dynamic environments. His D* algorithm has been used in unmanned ground vehicles, unmanned air vehicles, planetary rover prototypes, and indoor robots. Dr. Stentz and his students pioneered the use of market techniques to control a team of robots to provide fault tolerance, opportunistic optimization, and robustness to changing conditions, new tasks, and unexpected outcomes. He has transferred robotics technology to industry, by automating harvesting and spraying operations for agriculture, mass excavation for surface mining, continuous mining for underground mining, and inspection tasks for nuclear facilities.

Dr. Stentz has served on the editorial board or program committee for ICRA, AAAI, IAS, SPIE, and IFAC. He has executed projects for DARPA, NASA, ARL, DOE, USBM, NSF, Caterpillar, Boeing, New Holland, Westinghouse, General Dynamics, Joy Mining Machinery, and Deere & Company. He has over one hundred journal articles, conference and workshop papers, books, technical reports, and patents to his credit. Dr. Stentz is the recipient of the 1997 Alan Newell Award for Research Excellence.

Dr. David Wettergreen's research area is robotic exploration and spans concept formulation through system synthesis to field experimentation. He addresses exploration underwater, on the surface, and in air and space, and in the necessary ingredients of perception, planning, execution and control for robot autonomy. He is currently leading robotics research for the Life in the Atacama investigation using Hyperion, a solar-powered rover for life seeking. Hyperion has demonstrated 24-hour sun-synchronous navigation on Devon Island in the Canadian arctic and an instance of one-command, one-kilometer autonomous traverse in the Atacama Desert of Chile.

Dr. Wettergreen previously established a research program in underwater robotics at the Australian National University; the Kambara project investigates learning and adaptive methods of vehicle control and sensor-based guidance underwater. Dr. Wettergreen held an NRC Research Associateship at NASA Ames Research Center, were he developed a behavior-based control architecture for planetary rovers and conducted field experiments with new techniques for visual servo-control with the Marsokhod rover. He developed telepresence interfaces for the Nomad robot for its 200km trek in the Atacama Desert. In his doctoral research he investigated control and planning for legged robots through development of three systems: Ambler, a legged robot for Mars exploration, and Dante I & II, rappelling robots for volcano exploration. That research developed and demonstrated a hybrid control architecture for reactive behaviors and deliberative guidance and involved extensive field experimentation at sites including Mount Spurr, Alaska and Mount Erebus, Antarctica.